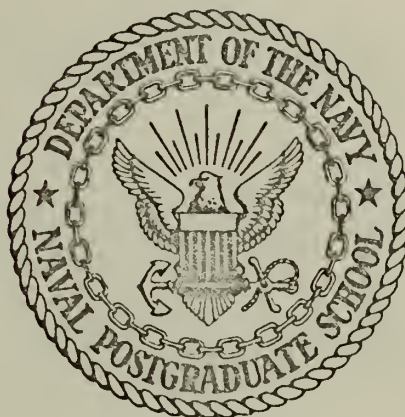


A SYMBOLIC SENSITIVITY COMPUTATION ALGORITHM

John Hale Daniel

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

A Symbolic
Sensitivity Computation Algorithm

by

John Hale Daniel

Thesis Advisor:

Shugar Chan

December 1972

T1530 2

Approved for public release; distribution unlimited.

A Symbolic
Sensitivity Computation Algorithm

by

John Hale Daniel
Lieutenant, United States Navy
B. S. E. E., University of Washington, 1964

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1972

ABSTRACT

A study and pursuant development is described of a digital computer program for the computation of transfer function sensitivities in a symbolic form suitable for storage and subsequent repetitive numerical evaluation over a range of frequencies or parameter values. The algorithm is implemented in conjunction with the Network Analysis for Systems Applications Program (NASAP) version developed by R. S. Schwartz at Northeastern University. Several example problems, as well as suggestions for possible improvement of the algorithm and its extrapolation into the areas of optimization and automatic design, are included.

TABLE OF CONTENTS

I. INTRODUCTION -----	4
A. STATEMENT OF THE PROBLEM -----	4
B. DISCUSSION OF THE NASAP PROGRAM -----	5
C. A COMPARISON OF TWO FAVORABLE TOPOLOGICAL METHODS -----	7
D. SUMMARY -----	9
II. DEVELOPMENTAL PROCEDURE -----	10
III. USER'S RULES -----	19
IV. INTERACTIVE APPLICATIONS -----	21
V. DISCUSSION AND RECOMMENDATIONS FOR FURTHER IMPROVEMENT -----	22
APPENDIX A SAMPLE PROBLEMS -----	24
APPENDIX B GUIDE FOR THE ALLOCATION OF CORE STORAGE -----	37
COMPUTER PROGRAM LISTING -----	38
BIBLIOGRAPHY -----	53
INITIAL DISTRIBUTION LIST -----	55
FORM DD 1473 -----	56

I. INTRODUCTION

A. STATEMENT OF THE PROBLEM

The purpose of this research was twofold: (1) to investigate the feasibility of providing for the automatic digital machine computation of transfer function sensitivities in a symbolic form suitable for storage and subsequent numerical evaluation over a range of frequencies and parameter values or for use by a minimization/optimization procedure as a readily available source of gradient data, and (2) to implement such a program, if feasible, in conjunction with the Network Analysis for Systems Applications Program (NASAP).

There were formidable obstacles encountered in the course of the research which may best be illustrated by a preliminary examination of the computer core storage required for implementation of the proposed procedure if it were meshed with an existent circuit analysis program such as NASAP. The NASAP program is capable of manipulating up to 10,000 independent loop products (generated through use of topological flowgraph techniques), once as a numerator polynomial in s and once as a denominator polynomial; that is, by iterative use of a single vector variable, LOOP, dimensioned to 10,000. Each loop product may represent from one to twenty-three constituent elements, and the analytical expression for the partial derivative of a quotient involves four separate components - numerator polynomial, denominator polynomial, partial derivative of the numerator and partial derivative of the denominator. The implication, then, is a storage allocation requirement of four vector variables each of

dimension 230,000 words (920K bytes). This, of course, would provide for the sensitivity with respect to a single parameter; if the gradient is desired the requirement for the two partial derivative variables would be multiplied by a factor of 23 again, for a total of 11,500,000 words. Consider now an even more staggering statistic. Since the NASAP program, like most programs written in the FORTRAN language, assigns only one alphanumeric character to a word (four bytes in the IBM 360/67), there would be an additional factor of four; three for the element identifying characters and one for a code to indicate the algebraic formulation required. This initial examination, then, would indicate the need for an impossibly high storage allocation of 46,000,000 words (184,000K bytes!). Some degree of success was achieved in the reduction of this figure to a manageable size, but at the expense of limitations on the original objective. The extent of the progress and associated restrictions will become clear in later sections.

B. DISCUSSION OF THE NASAP PROGRAM

Prior to discussing the sensitivity algorithm in any detail, it would perhaps benefit the reader to summarize the characteristics and capabilities of NASAP since it is upon this foundation that the proposed technique is firmly based.

The revised version of NASAP is a digital computer program for the flowgraph analysis of lumped, linear active electrical networks designed to provide any desired circuit transfer function in either symbolic or numerical form. The program utilizes the "mixed" method to generate circuit equations, i. e., operations are performed on a tree-branch, link-branch topology so that Kirchhoff's voltage law may be utilized to

express each link voltage in terms of tree voltages and Kirchoff's current law may be used to express each tree current in terms of link currents. These relationships, in association with volt-ampere and source dependency relationships, comprise a formulation defining a canonical form called a primitive flowgraph [3].

The user of NASAP prepares a circuit for analysis by: (1) converting to a linear model if necessary, (2) labeling each circuit element by type (R,L,C,V,I) and a two digit number which uniquely identifies it, and (3) numbering each node as desired except for the ground node which must be designated node 1. The user then inputs this connectivity data in the following format:

ELEMENT	FROM	TO	ELEMENT	CONTROLLING PARAMETER
NAME	NODE	NODE	VALUE	(For controlled source)

Following this are coded sets of data describing the transfer functions desired, in the form of the ratio of a response variable (voltage or current associated with some circuit element) to an excitation variable (voltage or current), and the forms of program output desired. The NASAP tree-link hierarchy [20] and the order in which the elements were input to the program determines which elements are placed in the tree, thus providing the user with some flexibility regarding its choice. Given the topology, the machine then forms the fundamental outset matrix and from this constructs the primitive flowgraph.

A transfer function between any two nodes of the primitive flowgraph may be obtained by adding a "closure" [20] and solving the resulting "closed" graph by application of Mason's gain formula in the following manner. If the transfer function between node j and node k of a flowgraph is equal to T_{jk} and if the graph is then closed by the addition of

a branch from k to j whose transmission gain is equal to the reciprocal of T_{jk} , then the loop gain with the added branch is equal to unity and the determinant of the flowgraph vanishes. This reduces the solution of T_{jk} to the problem of summing loop-set transmittance products, a procedure highly suitable to machine solution. Readers desiring more detailed information concerning this procedure are directed to Ref. 20.

The procedures utilized by NASAP for labeling, sorting and storing each loop-set consist primarily of:

- (1) As each loop-set is determined by the search routine, a signed integer label is generated from which the loop-set may be uniquely regenerated.
 - (2) Loop-sets with the same label but opposite sign are eliminated to reduce storage requirements.
 - (3) Loop-sets are sorted and stored in order of decreasing magnitude.
 - (4) The completed loop-set array represents the symbolic solution of either the numerator or denominator polynomial in coded form.
- From successive solutions the transfer function may be evaluated either numerically or symbolically by decoding and summing each loop.

C. A COMPARISON OF TWO FAVORABLE TOPOLOGICAL METHODS

A topological method other than the flowgraph procedure just described is the so-called "k-tree" approach [6] which is based on Maxwell's topological formulas for node voltage equations. That is, transfer functions are derived from the evaluation of determinants formed from circuit nodal equations; however, they may equivalently be formed in a topological fashion by summing all tree admittance products and appropriate 2-tree, 3-tree, ... k-tree admittance products.

In both methods, k-tree and loop-set, the solution of passive circuits is obtained as a sum of linear products of the network parameters, thus eliminating the requirement for operations with polynomials, and facilitating machine evaluation of symbolic solutions. In addition, the two methods compare favorably with respect to the number of loop-transmittance products and tree-admittance products for a given circuit, since the maximum number of trees possible for a given topology is only one greater than the maximum number of loop-sets contained in the corresponding primitive flowgraph [20]. Each tree-admittance product is unique; however, in the flowgraph method several loop-sets are frequently found to have identical transmittance product magnitudes, although only one will remain after cancellation. Thus each remaining product is unique. A primary advantage of both procedures is this feature which obviates the evaluation of zero terms, as they would be if the system determinant were evaluated by matrix algebra.

Although the two methods appear equal from the above discussion, a very important difference arises in the solution of active networks since the k-tree approach requires the graph to be modified and divided into separate current and voltage graphs from each of which the complete k-trees must be determined, whereas in the loop-set analysis the procedure is invariant for passive or active networks.

Another significant disadvantage of the k-tree analysis is its dependence upon the particular transfer function to be evaluated, i.e., impedance, admittance, gain, etc., each requiring different types of k-tree searches. Thus, the number of k-tree searches will vary with the number of transfer functions as well as the type desired. Therefore, the flowgraph

method appears to have a distinct advantage in this respect, since the procedure remains identical regardless of the number or types of transfer functions. For these reasons the flowgraph approach was chosen for implementation in NASAP.

D. SUMMARY

It was the objective of this study to determine the feasibility of utilizing the loop-set symbolic form of the transfer function originated in the manner described in (B) above as a basis for computing and storing sensitivity information in similar symbolic form, thereby vastly simplifying the repetitive computation of accurate network sensitivities and rendering such solutions free of the difficulties involved with differential, or numerical techniques.

II. DEVELOPMENTAL PROCEDURE

Utilizing the revised version of NASAP developed for the PDP-10 and CDC-3300 computers by R. S. Schwartz [20], adaptation to the IBM 360/67 computer at the Naval Postgraduate School, Monterey was accomplished by: (1) altering program statements necessary to reconcile the difference in word lengths between the two types of computers, (2) making various syntax changes necessitated by technical differences in the implementation of FORTRAN on the different machines, and (3) changing the modular form and adding program statements necessary to provide for the execution of the algorithm as intended by the author. Details of the conversion are apparent from comparison of the program listing available at the W. R. Church Computer Center, USNPGS, Monterey, California with that of Ref. 20. The listing found in the appendix of this thesis contains only the NAS1, NAS4, and SENS subroutines. The first two are included since extensive modifications were incorporated in the course of implementing the sensitivity feature, and the latter because it is entirely new.

As previously described, the NASAP program has available, in symbolic form, the uncanceled loop products created from a topological flowgraph analysis of a given circuit. It was from this base that the writer devised a program to compute circuit transfer function sensitivities in an algorithm developed as follows.

Since, in subroutine NAS4 of the original program, there implicitly exists a requested transfer function,

$$T(s) = K \frac{N(s)}{D(s)} \text{ in which,}$$

$$N(s) = b_0 s^n + b_1 s^m + \dots + b_i s^p + \dots + b_{LPN} s^q$$

$$D(s) = a_0 s^{n'} + a_1 s^{m'} + \dots + a_i s^{p'} + \dots + a_{LPD} s^{q'}$$

where $n, m, p, q, n', m', p', q', \dots$ are integer exponents of the Laplace variable s , not sorted into any ascending or descending order and with repetitions possible; and where a_i, b_i represent the i^{th} loop products (algebraic product/quotients of circuit parameters in symbolic form); and where LPD is the number of uncanceled loop products in the denominator polynomial of $T(s)$ and LPN the number in the numerator. Each loop product is actually determined from one 32-bit computer word, wherein the presence (1) or absence (0) of each element is indicated by the corresponding bit value. This, of course limits the number of circuit parameters which can be accommodated by the program to $NBITS-1 = 31$, although in the current version this upper limit is set to 23 due to other considerations.

It was desired, then, to compute the sensitivity of $T(s)$ in symbolic form, to a specified circuit parameter P_k . Since sensitivity is essentially a rate-of-change, it follows that the transfer function sensitivity may be defined as:

$$S_{P_k}^T = \frac{\partial T(s)}{\partial P_k} = \frac{\partial (N(s)/D(s))}{\partial P_k} = \frac{D(s) \cdot \frac{\partial N(s)}{\partial P_k} - N(s) \cdot \frac{\partial D(s)}{\partial P_k}}{D^2(s)} \quad (1)$$

To implement this fundamental approach utilizing the data available in NAS4, the following simple algorithm was developed, (See Flow-Chart, Fig. 1):

(I) Observe the j^{th} factor of the i^{th} loop product, b_{ij} , in the numerator of T .

Is b_{ij} in the numerator of the i^{th} loop product, b_i ?

YES: (a) Is this factor the parameter P_k ?

YES: (i) Set flag to indicate that b_i contains P_k .

(ii) Form: $\left(\frac{\partial N}{\partial P_k}\right)_{ij} = 1$

NO: Form: $\left(\frac{\partial N}{\partial P_k}\right)_{ij} = b_{ij}$

(b) $N_{ij} = b_{ij}$

NO: (a) Is this factor P_k ?

YES: (i) Set flag

(ii) Change sign of $\left(\frac{\partial N}{\partial P_k}\right)_i$

(iii) Form: $\left(\frac{\partial N}{\partial P_k}\right)_{ij} = 1/(b_{ij})^2$

NO: Form $\left(\frac{\partial N}{\partial P_k}\right)_{ij} = 1/b_{ij}$

(b) $N_{ij} = 1/b_{ij}$

(II) Repeat step (I) for the $(j+1)^{th}$ factor of the i^{th} loop product.
Continue until all factors are exhausted ($j > NFN(i)$, where $NFN(i)$ represents the number of factors in the i^{th} loop product of the numerator polynomial).

(III) Does $\left(\frac{\partial N}{\partial P_k}\right)_i$ contain P_k ? (is flag set to 1?)

YES: Continue

NO: Set $\left(\frac{\partial N}{\partial P_k}\right)_i = 0$

IV) (A) $i = i + 1$

(B) Repeat (I) (II), and (III) for the $(i+1)^{th}$ loop product.

Continue until all loops are exhausted ($i > LPN$).

(V) Repeat the entire procedure beginning with step (I) for the denominator of $T(s)$.

At this point the sensitivity is implicitly available in the desired form (eqn. 1), where:

$$D = \sum_{i=1}^{LPD} \prod_{j=1}^{NFD(i)} D_{ij} s^{EXPDS(i)}$$

$$N = \sum_{i=1}^{LPN} \prod_{j=1}^{NFN(i)} N_{ij} s^{EXPNS(i)}$$

$$\frac{\partial D}{\partial P_k} = \sum_{i=1}^{LPD} \prod_{j=1}^{NFD(i)} \left(\frac{\partial N}{\partial P_k} \right)_{ij} s^{EXPDS(i)}$$

$$\frac{\partial N}{\partial P_k} = \sum_{i=1}^{LPN} \prod_{j=1}^{NFN(i)} \left(\frac{\partial N}{\partial P_k} \right)_{ij} s^{EXPNS(i)}$$

$NFN(i)$ = Number of factors in the i^{th} loop product of numerator polynomial

$NFD(i)$ = Number of factors in the i^{th} loop product of denominator

$EXPNS(i)$ = Exponent of s in the i^{th} loop product of numerator

$EXPDS(i)$ = Exponent of s in the i^{th} loop product of denominator

One feature of the programming which became necessary in order to reduce storage requirements for the symbolic sensitivity component variables $SNUM$, $SDEN$, $PARTLN$, and $PARTLD$ was a packing scheme which in effect reduced vector sizes by a factor of one-fourth. In the NASAP program, as in most FORTRAN programs, each alphanumeric character is represented by a four-byte word, wasting a great deal of core storage since one byte is sufficient for one character. This exemplifies one of the most unattractive aspects of the FORTRAN language, especially when dealing with character strings or literals as in the case of the NASAP symbolic version.

A remedy for the problem which was utilized in this program is a very short but efficient packing subroutine which has four arguments:

- (1) a four-byte variable to be "packed"
- (2) an integer constant specifying the byte number of the variable in (1) to be assigned the value of the packing variable
- (3) packing variable
- (4) an integer constant specifying the byte number of the variable in (3) which is to be assigned to the variable in (1)

Upon calling the pack subroutine, dummy variables, locally declared LOGICAL*1 (one byte), receive data passed from the specified byte of the packing variable, assign this to the specified byte of the variable to be packed, and then return this new "packed" version to the calling program. In this manner the sensitivity component variables were packed with parameter name (e.g. L01, R04, etc.) in bytes one through three, and byte number four was packed with a code number to indicate the form in which the parameter should appear. This code appears below:

- 0 -- Numerator or denominator = z , partial = z
- 3 -- Numerator or denominator = z , partial = 1
- 4 -- Numerator or denominator = z , partial = 0
- 6 -- Numerator or denominator = $1/z$, partial = $1/z$
- 7 -- Numerator or denominator = $1/z$, partial = $-1/z^2$
- 9 -- Numerator or denominator = $1/z$, partial = 0

With this scheme the program was rendered capable of processing networks of up to 200 independent loops to provide for sensitivity with respect to a single parameter, or by successive solutions to provide for any or all sensitivities desired. With this loop-set limit, the entire program, including NASAP requires 180K of core. A guide for core storage allocation may be found in the appendix for circuits of greater complexity.

Subsequent to the packing operation, data is transferred to a new subroutine SENS for manipulation into the desired form and for numerical evaluation. In this operation it was found impractical, of course, to permit each product element to occupy a separate storage location in the computer, since this would essentially require dimensions on the order of the square of the maximum number of loop products. Therefore, numerical evaluation of the constituent components of the sensitivity are recursively computed and simultaneously combined at this point in the program to form the numerator and denominator arrays, PART1 and PART2. Eliminating all zero vectors and combining like powers of s reduced storage requirements for this operation essentially to twice the order of the denominator polynomial.

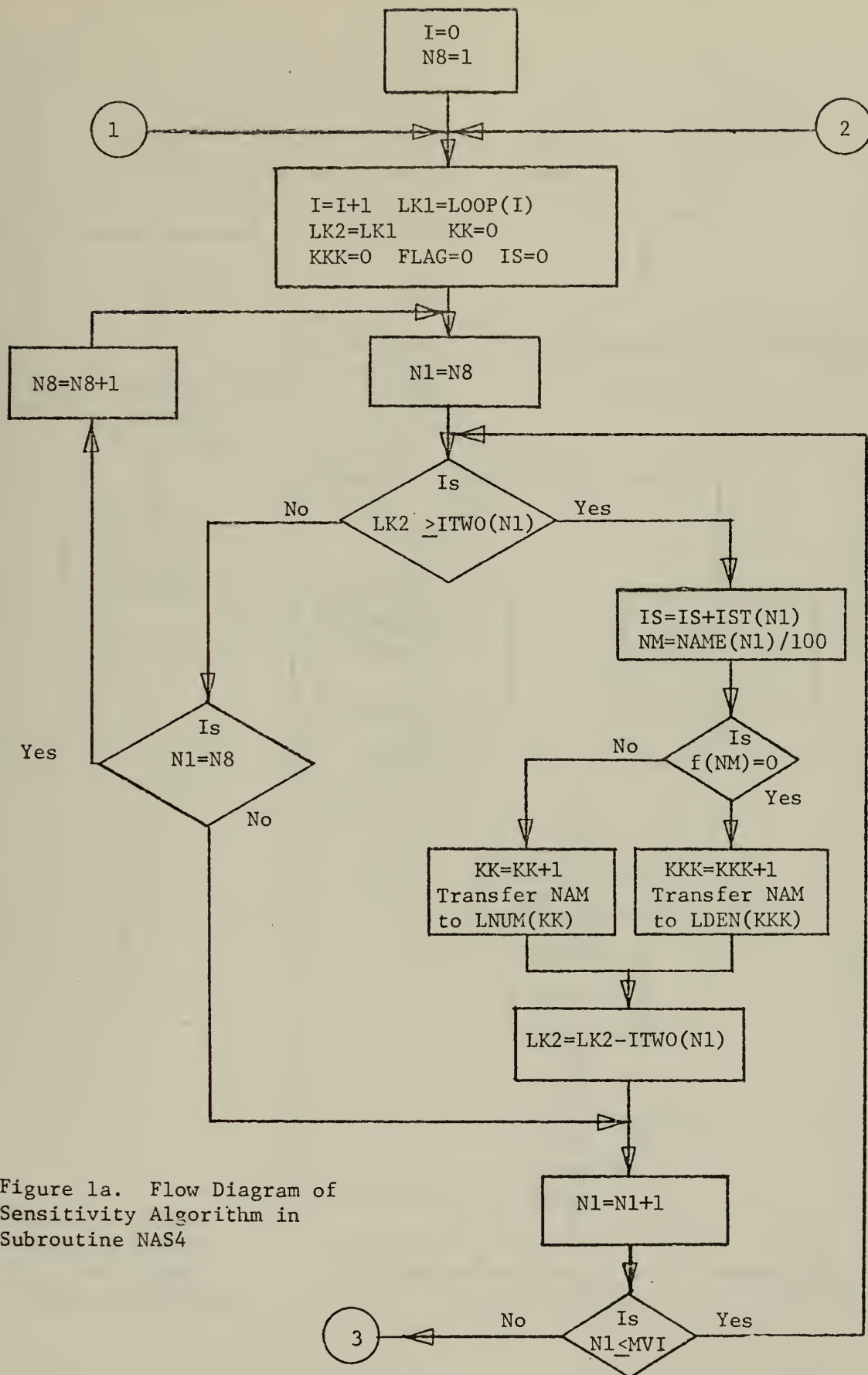


Figure 1a. Flow Diagram of Sensitivity Algorithm in Subroutine NAS4

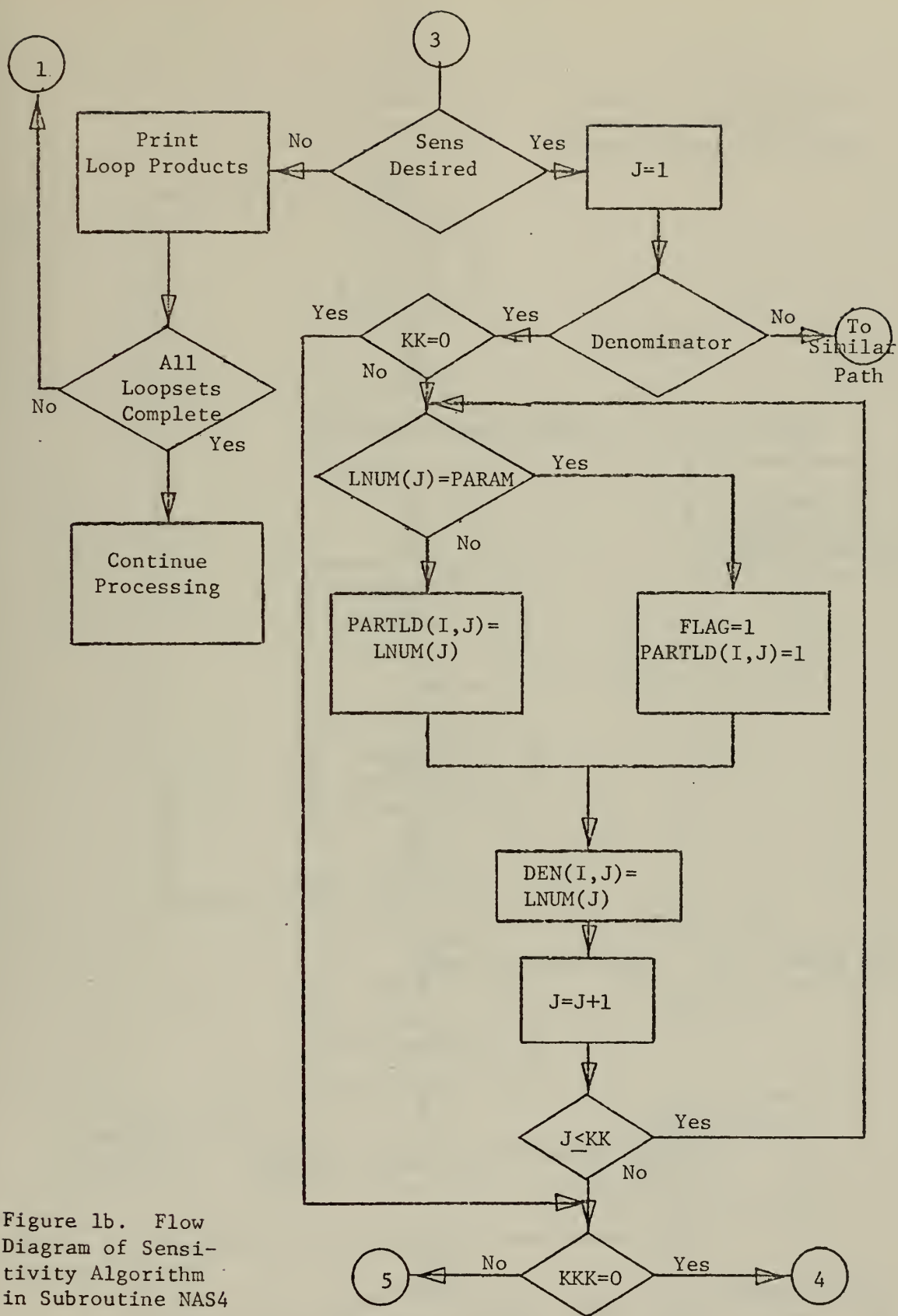
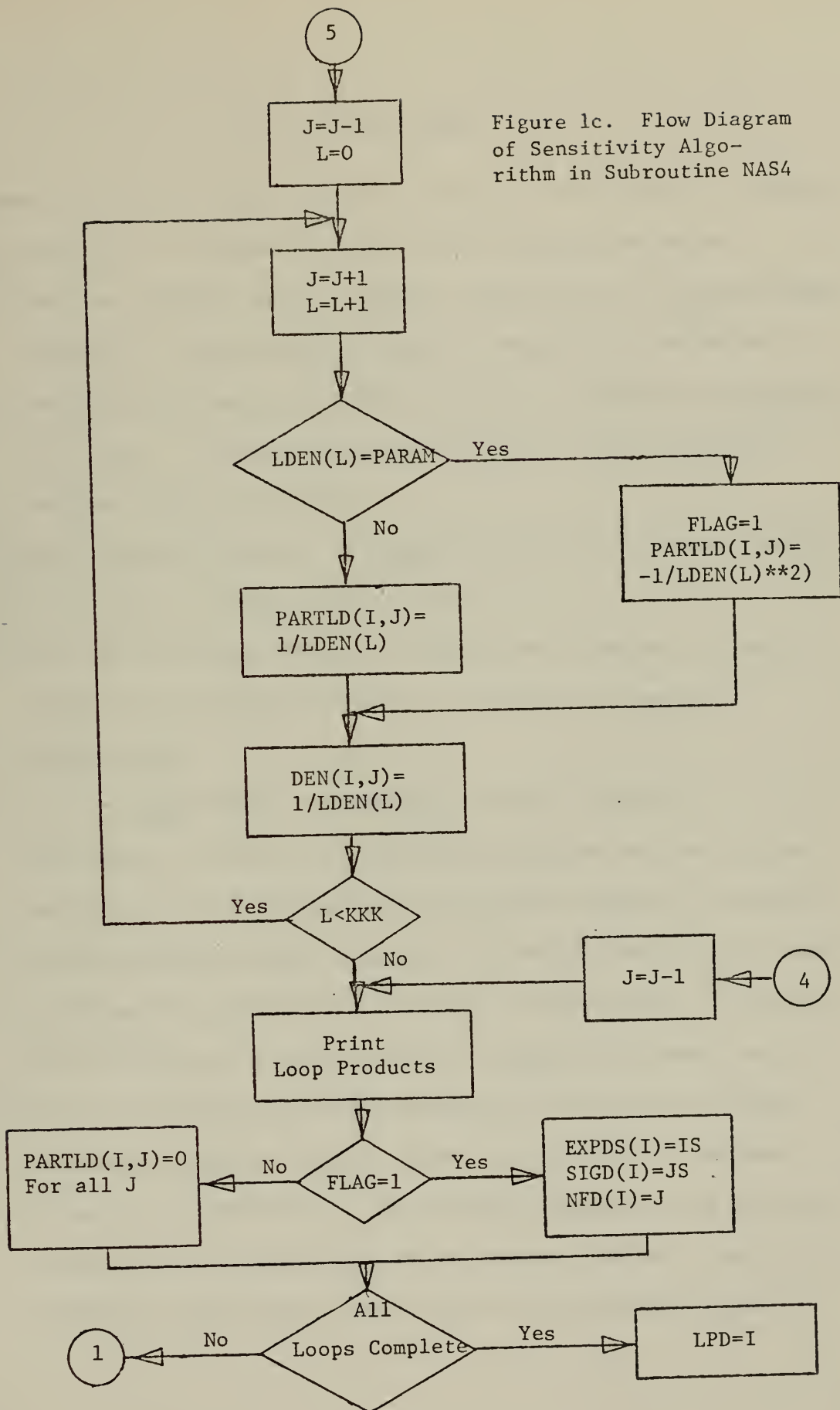


Figure 1b. Flow Diagram of Sensitivity Algorithm in Subroutine NAS4



III. USER'S RULES

Users of the sensitivity subroutine must be familiar with the simple NASAP coding rules contained in Refs. 1 and 2 as modified below:

- (1) Only one "OUTPUT" and/or "ROOTS" transfer function request should be made in a given problem since the sensitivity subroutine is designed to store data relevant to only one numerator polynomial at a time. Of course, the user may wish to follow one set of problem cards with another.
- (2) The "SYMBOLIC SOLUTION" card must be inserted following "EXECUTE".
- (3) No "MODIFY" requests are permitted.
- (4) The card following "SYMBOLIC SOLUTION" must contain the words (only the underlined portions are essential for machine recognition);

SENSITIVITY PARAMETER=L02 (or R01, C09, etc.)

Any number of blanks or identifying characters may be included so long as the underlined portions appear somewhere in columns 1-80 in the order shown. Notice that the parameter (with respect to which the sensitivity is desired) is indicated by the consecutive characters, (=), (R, L, or C), (two digit element number).

- (5) If it is desired to have the sensitivity magnitude and phase evaluated, the next card in sequence must contain a maximum frequency in columns 1-11 and a minimum frequency (not zero) in columns 21-31, preferably both in standard FORTRAN formal E11.4, although normal floating-point form is satisfactory if all

significant digits (plus sign) remain within the specified 11 columns. Specifying these frequencies limits will result in a print-out of the sensitivity magnitude and phase at 200 evenly distributed frequencies in the interval.

(Caution: If the user omits this option and desires to follow the current problem deck with others, a blank card must be inserted in place of the frequency interval card).

An interested user may provide for the plotting of the generated magnitude and phase data by changing the variables RMAGN, ANGLE, and FREQ into array variables in the SENS subroutine, then passing them to an existing library plotting subroutine. This will, of course, considerably increase the core storage required for the program.

IV. INTERACTIVE APPLICATIONS

The NASAP program, in conjunction with the sensitivity subroutine, may be easily adapted for use under a time-sharing system such as CP/CMS with only a few simple statement changes which depend upon the user's choice of file structure. One choice successfully used at the Naval Postgraduate School is to remove the four initial statements: (1) CALL NASAP, (2) STOP, (3) END, and (4) SUBROUTINE NASAP. It is possible, then, to "offline read" the program into CP in the following modular form:

<u>FILE</u>	<u>PROGRAM CONTENT</u>
RSS	Program RSS (Driving program)
NAS1	NAS1, FUNCTION ITRAN
NAS2	NAS2
NAS3	NAS3
NAS4	NAS4, SUBROUTINE PACK
NAS5	NAS5
SENS	SENS, BLOCK DATA

Under CP/CMS, the entire program can then be implemented by executing "RSS", with data being input at the terminal, or if the three "READ" statements in NAS1 are changed appropriately, data may be read from a separate file so defined.

V. DISCUSSION AND RECOMMENDATIONS FOR FURTHER IMPROVEMENT

As the reader may have gathered, the feasibility of computing sensitivities of a transfer function to all circuit parameters and storing them in symbolic form appears doubtful under present storage limitations. What follows, however, is a recommendation for further reducing storage requirements somewhat while maintaining the symbolic concept. If only a single numerical sensitivity were desired, it would of course be considerably more efficient to recursively evaluate the sensitivity from successive loop-sets with their appropriate values first as given, then in partial derivative form. This procedure would eliminate most of the storage problems. The singular advantage of the symbolic approach, however, is that evaluation of the sensitivity function at various frequencies, or for various parameters, is a very simple matter of substitution, rather than the use of time-consuming iterative or numerical techniques which may pose stability or accuracy difficulties.

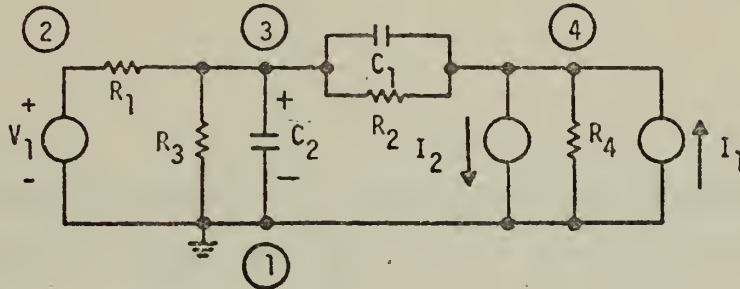
It is apparent that substantial changes are indicated in the proposed algorithm (or perhaps only the techniques for implementing the algorithm which will permit the sensitivity subroutine to cope with circuits of greater complexity by means of a reduction in the large storage requirements. However, it must be remembered that the resulting exponential increase in the number of independent loops with respect to a concomitant increase in the number of nodes [20] will in itself limit the complexity of topological/flowgraph methods, particularly in maximal-planar networks. Indeed, the difficulties of dimensionality plague all circuit analysis programs.

One suggested method of reducing storage requirements would be to extend the previously mentioned concept of utilizing individual bits of the computer word to represent constituent elements of the network. Since in this case it is necessary to know which of six mathematical formulations is appropriate for each element of the numerator and denominator polynomials, instead of only two formulations as in the case of transfer function loops, it would appear that a scheme requiring only three variables ($2^3 = 8$ possibilities), for a numerator and denominator each dimensioned to the maximum number of loops (10,000 in the present version) would suffice to allow sensitivities to be computed and stored in symbolic form. This would represent a substantial reduction from the 460,000 that are presently required for a 10,000-loop circuit to accomplish the same task. This suggestion, in effect, offers an execution time-vs-storage tradeoff, since a coding/decoding algorithm must be developed which will intrinsically involve a tremendous number of manipulations. In addition it would require an almost total rewrite of the FORTRAN programming, particularly that portion meshed with the NASAP NAS4 subroutine, and it may very well lead to difficulties not apparent without further investigation.

APPENDIX A SAMPLE PROBLEMS

(I) Sample problem No. 1 - Emitter Follower

(a) Emitter follower linear model:



(b) NASAP coding:

```

NASAP PROBLEM NO. 1
*EMITTER FOLLOWER
V1 1 2 1
I1 1 4 1
R1 2 3 100
R2 3 4 1M
R3 3 1 1.5K
R4 4 1 100
C1 3 4 6PF
C2 3 1 200PF
I2 4 1 0.04VC2
OUTPUT
VR4/VV1
EXECUTE
SYMBOLIC SOLUTION
SENSITIVITY PARAMETER=R04
    
```


(c) Sensitivity output for emitter follower:

NUMERATOR POLYNOMIAL : SENSITIVITY OF VR04/VV01 TO R04

0.6944E 40 TIMES S** 0

0.1042E 31 TIMES S** -1

0.6667E 17 TIMES S** -2

DENOMINATOR POLYNOMIAL : SENSITIVITY

0.2778E 19 TIMES S** 2

0.3473E 31 TIMES S** 1

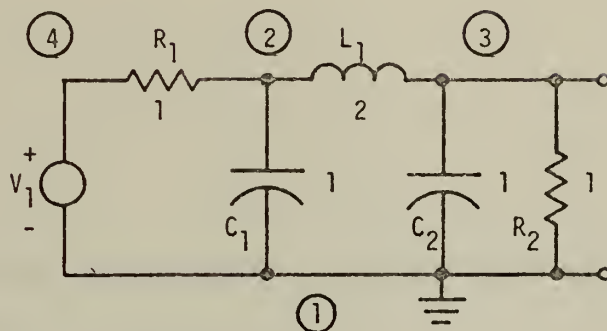
0.1085E 43 TIMES S** 0

0.1111E 30 TIMES S** -1

0.2845E 16 TIMES S** -2

(II) Sample Problem No. 2 - Butterworth Filter

(a) Original linear circuit:



(b) NASAP coding:

```
NASAP PROBLEM
*THIRD ORDER BUTTERWORTH
R1 4 2 1
R2 3 1 1
C1 2 1 1
L1 2 3 2
C2 3 1 1
V1 1 4 1
OUTPUT
VR2/VV1
EXECUTE
SYMBOLIC SOLUTION
SENSITIVITY PARAMETER=R02
```


(c) Sensitivity output for Sample Problem No. 2 - Butterworth

Filter:

NUMERATOR POLYNOMIAL : SENSITIVITY OF VR02/VV01 TO R02

0.5000E 00 TIMES S** 0

0.5000E 00 TIMES S** -1

0.2500E 00 TIMES S** -2

DENOMINATOR POLYNOMIAL : SENSITIVITY

0.1000E 01 TIMES S** 4

0.4000E 01 TIMES S** 3

0.8000E 01 TIMES S** 2

0.1000E 02 TIMES S** 1

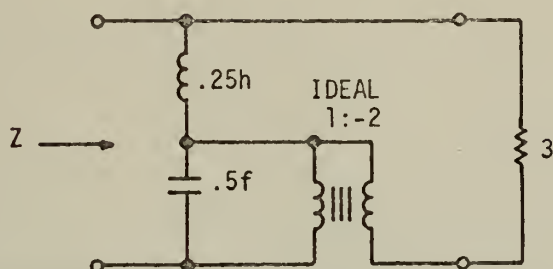
0.8000E 01 TIMES S** 0

0.4000E 01 TIMES S** -1

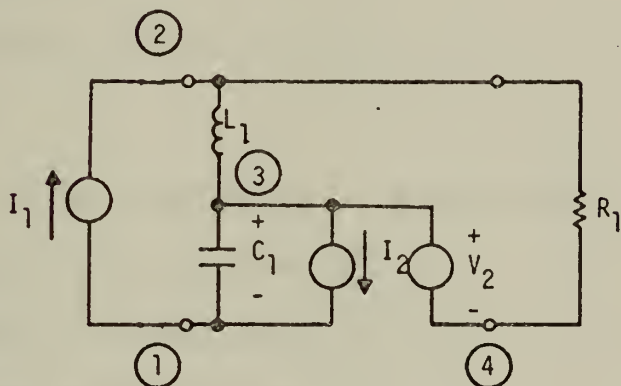
0.1000E 01 TIMES S** -2

(III) Sample Problem No. 3 - Transformer circuit

(a) Original circuit with ideal transformer:



(b) Linear model:



(c) NASAP coding:

```

NASAP PROBLEM
*INPUT IMPEDANCE
I1 1 2 1
R1 2 4 3
C1 3 1 0.5
L1 2 3 0.25
I2 3 1 -2 IR1
V2 4 3 -2 VC1
OUTPUT REQUEST
VI1/II1
EXECUTE
SYMBOLIC SOLUTION
SENSITIVITY PARAMETER=L01
.1592E 03 .1592E 00
    
```


(d) Sensitivity output for Sample problem No. 3 - Transformer
circuit:

NUMERATOR POLYNOMIAL : SENSITIVITY OF VI01/II01 TO L01

0.1000E 01 TIMES S** 1
-0.3333E 00 TIMES S** 0
0.2778E-01 TIMES S** -1

DENOMINATOR POLYNOMIAL : SENSITIVITY

0.1778E 01 TIMES S** 2
0.2667E 01 TIMES S** 1
0.1444E 01 TIMES S** 0
0.3333E 00 TIMES S** -1
0.2778E-01 TIMES S** -2

(e) Sensitivity magnitude and phase output for Sample Problem

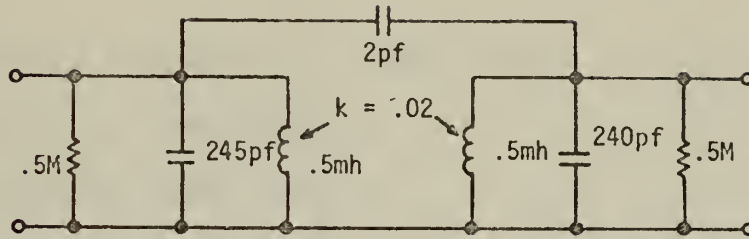
No. 3 - Transformer circuit:

EVALUATION OF SENSITIVITY

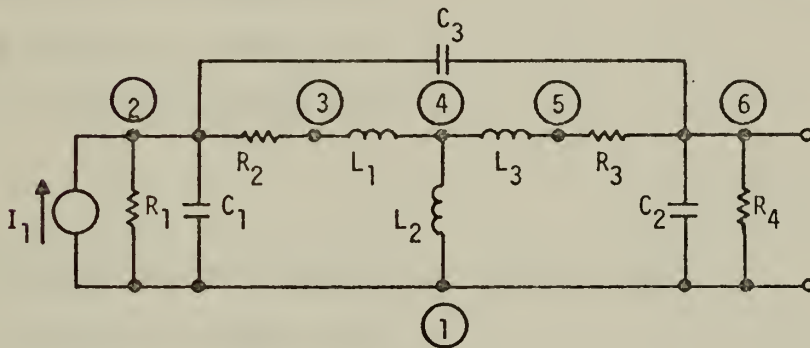
FREQUENCY	MAGNITUDE	PHASE(IN RADIAN)
0.1592E 00	0.4352E 00	0.1763E 00
0.9584E 00	0.9268E-01	-0.1267E 01
0.1758E 01	0.5082E-01	-0.1405E 01
0.2557E 01	0.3498E-01	-0.1457E 01
0.3356E 01	0.2666E-01	-0.1484E 01
0.4155E 01	0.2154E-01	-0.1501E 01
0.4954E 01	0.1806E-01	-0.1512E 01
0.5754E 01	0.1556E-01	-0.1520E 01
0.6553E 01	0.1366E-01	-0.1526E 01
0.7352E 01	0.1218E-01	-0.1531E 01
0.8151E 01	0.1098E-01	-0.1535E 01
0.8950E 01	0.1000E-01	-0.1538E 01
<hr/>		
0.6250E 02	0.1432E-02	-0.1566E 01
0.6330E 02	0.1414E-02	-0.1566E 01
0.6409E 02	0.1397E-02	-0.1566E 01
0.6489E 02	0.1380E-02	-0.1566E 01
0.6569E 02	0.1363E-02	-0.1566E 01
0.6649E 02	0.1346E-02	-0.1566E 01
0.6729E 02	0.1330E-02	-0.1566E 01
0.6809E 02	0.1315E-02	-0.1567E 01
0.6889E 02	0.1300E-02	-0.1567E 01
0.6969E 02	0.1285E-02	-0.1567E 01
<hr/>		
0.1480E 03	0.6049E-03	-0.1569E 01
0.1488E 03	0.6016E-03	-0.1569E 01
0.1496E 03	0.5984E-03	-0.1569E 01
0.1504E 03	0.5952E-03	-0.1569E 01
0.1512E 03	0.5921E-03	-0.1569E 01
0.1520E 03	0.5890E-03	-0.1569E 01
0.1528E 03	0.5859E-03	-0.1569E 01
0.1536E 03	0.5828E-03	-0.1569E 01
0.1544E 03	0.5798E-03	-0.1569E 01
0.1552E 03	0.5768E-03	-0.1569E 01
0.1560E 03	0.5739E-03	-0.1569E 01
0.1568E 03	0.5709E-03	-0.1569E 01
0.1576E 03	0.5680E-03	-0.1569E 01
0.1584E 03	0.5652E-03	-0.1569E 01
0.1592E 03	0.5623E-03	-0.1569E 01

(IV) Sample Problem No. 4 - I.F. Transformer

(a) I.F. Transformer circuit:



(b) Linear model:



(c) NASAP coding:

```

NASAP PROBLEM
*I.F. TRANSFORMER
I1 1 2 1
R1 2 1 .5M
C1 2 1 245PF
R2 2 3 10
L1 3 4 .49MH
L2 4 1 .01MH
L3 4 5 .49MH
R3 5 6 10
C2 6 1 240PF
C3 2 6 2 PF
R4 6 1 .5M
OUTPUT
VR4/I11
EXECUTE
SYMBOLIC SOLUTION
SENSITIVITY PARAMETER=C03
    
```


(d) Sensitivity output for Sample Problem No. 4 - I.F.

Transformer:

NUMERATOR POLYNOMIAL : SENSITIVITY OF VR04/1101 TO C03

0.4720E 18 TIMES S** 5

0.4786E 16 TIMES S** 4

0.1293E 14 TIMES S** 3

0.4049E 10 TIMES S** 2

0.4725E 06 TIMES S** 1

0.2309E 02 TIMES S** 0

0.4000E-03 TIMES S** -1

DENOMINATOR POLYNOMIAL : SENSITIVITY

0.4760E 42 TIMES S** 6

0.4827E 40 TIMES S** 5

0.1304E 38 TIMES S** 4

0.4095E 34 TIMES S** 3

0.4805E 30 TIMES S** 2

0.2375E 26 TIMES S** 1

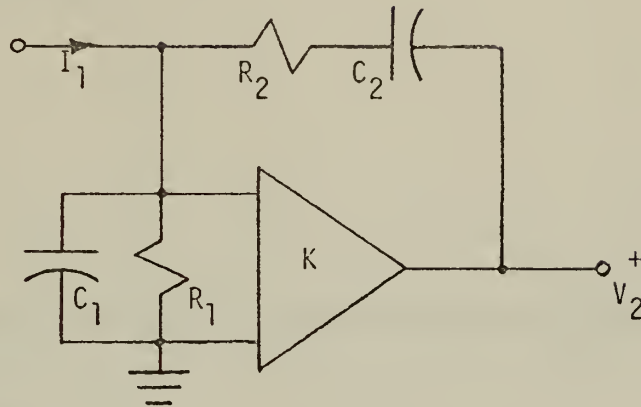
0.4266E 21 TIMES S** 0

0.4074E 15 TIMES S** -1

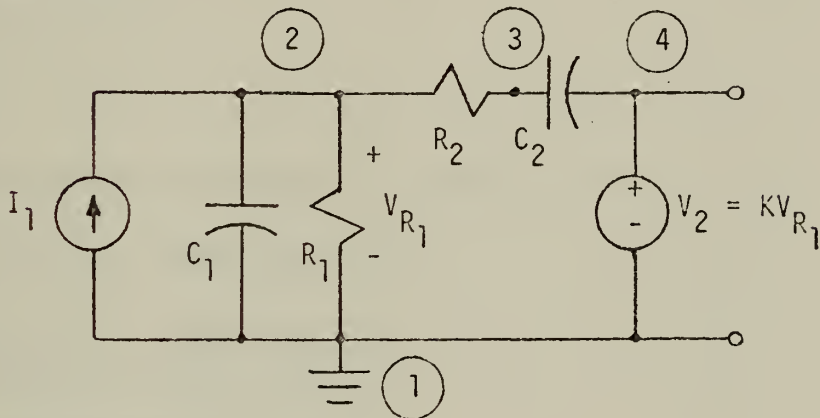
0.1000E 09 TIMES S** -2

(V) Sample Problem No. 5 - Wien-Bridge circuit

(a) Original circuit:



(b) Linear model:



(c) NASAP coding:

```

NASAP PROBLEM
*WIEN-BRIDGE CIRCUIT
I1 1 2 1
R1 2 1 1
C1 2 1 1
R2 2 3 1
C2 3 4 1
V2 1 4 1 VR1
OUTPUT
VV2/III1
EXECUTE
SYMBOLIC SOLUTION
SENSITIVITY PARAMETER=R02
    
```


(d) Sensitivity output for Sample Problem No. 5 - Wien-Bridge
circuit:

NUMERATOR POLYNOMIAL : SENSITIVITY OF VV02/1101 TO R02
IDENTICALLY ZERO

DENOMINATOR POLYNOMIAL : SENSITIVITY

0.1000E 01 TIMES S** 4

0.4000E 01 TIMES S** 3

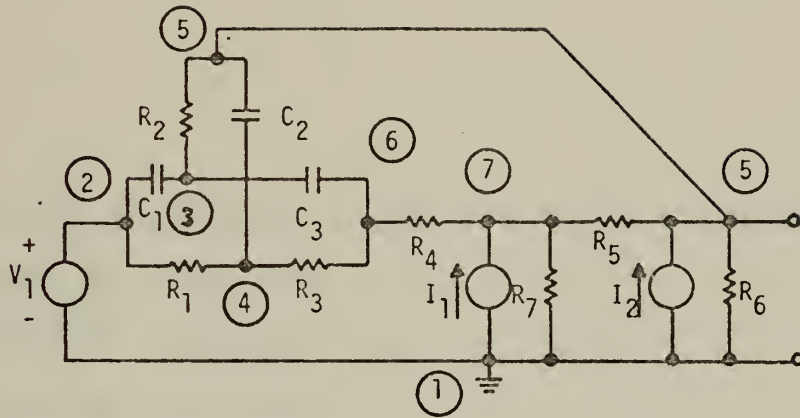
0.6000E 01 TIMES S** 2

0.4000E 01 TIMES S** 1

0.1000E 01 TIMES S** 0

(VI) Sample Problem No. 6 - High-Q Filter

(a) Filter circuit:



(b) NASAP Coding:

```

NASAP PROBLEM
*HIGH - Q FILTER
V1 1 2 1
R1 2 4 20K
R2 5 3 10K
R3 6 4 20K
R4 6 7 100
R5 7 5 100
R6 5 1 2K
R7 7 1 1M
C1 2 3 .02UF
C2 5 4 .04UF
C3 6 3 .02UF
I1 1 7 150 IR4
I2 1 5 150 IR5
OUTPUT
VR6/VV1
EXECUTE
SYMBOLIC SOLUTION
SENSITIVITY PARAMETER=C02
    
```


(c) Sensitivity output for Sample Problem No. 6 - High-Q filter:

NUMERATOR POLYNOMIAL : SENSITIVITY OF VR06/VV01 TO C02

-0.2684E 09 TIMES S** 6

0.3125E 12 TIMES S** 5

0.3125E 09 TIMES S** 4

0.1031E 06 TIMES S** 3

0.1250E 02 TIMES S** 2

0.5000E-03 TIMES S** 1

DENOMINATOR POLYNOMIAL : SENSITIVITY

0.1650E 24 TIMES S** 6

0.2869E 21 TIMES S** 5

0.1887E 18 TIMES S** 4

0.5643E 14 TIMES S** 3

0.6908E 10 TIMES S** 2

0.1575E 06 TIMES S** 1

0.1000E 01 TIMES S** 0

APPENDIX B GUIDE FOR THE ALLOCATION OF CORE STORAGE

<u>NUMBER OF LOOP-SETS POSSIBLE (VARIABLE DIMENSION)</u>	<u>PROGRAM CORE REQUIRED</u>
* 200	180K
300	205K
400	230K
500	255K
600	280K
700	305K
** 800	330K
900	355K
1000	380K
*** 5000	505K
10000	630K

* Corresponds approximately to a maximal planar network
of 5 nodes

** Corresponds approximately to a maximal planar network
of 6 nodes

*** Corresponds approximately to a maximal planar network
of 7 nodes

PROGRAM LISTING

```

SUBROUTINE NAS1
  INTEGER EOFCKF, SENSFL, PARAM
  DIMENSION INLST(80), NODE(21)
  COMMON/SEN/ PARAM(3), SENSFL, FRMAX, FRMIN
  COMMON/DATA/ INSR1(100), INSR2(24,6), ITSK1(15,4), ITSK2(3
1,6), ITSK3
5(15,6), ITSK4(10,5), ITSK5(9,10), IRAY(19,2), ISYMB(48),
6ITASK6(17,12), IOPRTN(3,2)
  COMMON IND, NASA, MTRAN(3), MD, IJTAG(15,2), IPRIN, Z(42
1), LL, MODNM(23), VALMOD(23,10), ILOOP(23), LI, LIC, M9, ZZ(23
2),
3NAM(23,3), ITAG(5,8), NMBR(6),
4M, LCONC(10,40), NAME(23,2), MVI, IST(40), INC(38,42), N1, NA
+, M2
  DATA IB/'      ' /, IE/1HE/
  FRMAX=0.0
  FRMIN=0.0
66 READ(5,101,END=998) INLST
65 DO 1 J=1,32
  IF(INLST(J)-IB)300,1,300
300 DO 2 I=1,3
  IF(INLST(J)-IOPRTN(1,1))2,350,2
350 IF(INLST(J+1)-IOPRTN(1,2))2,360,2
2 CONTINUE
  GOTO 3
360 GOTO(4,301,351),I
1 CONTINUE
  GO TO 66
351 GO TO (67,303,352,68,68),NASA
352 IPRIN=2
  IND=4
  WRITE (6,119)
  WRITE (6,116)INLST
  READ(5,101,END=1357) INLST
  IF(INLST(1)-IOPRTN(3,1))1357,1351,1357
1351 IF(INLST(2)-IE)1357,1352,1357
1352 SENSFL=1
  DO 1353 K=3,80
  IF(INLST(K)-ISYMB(46))1353,1355,1353
1353 CONTINUE
1355 KP1=K+1
  DO 1354 I=KP1,80
  IF(INLST(I).EQ.IB) GO TO 1354
1356 PARAM(1)=INLST(I)
  DO 1350 MM=1,10
  IF(INLST(I+1).EQ.ISYMB(MM)) GO TO 1361
1360 CONTINUE
1361 PARAM(2)=MM-1
  DO 1362 MM=1,10
  IF(INLST(I+2).EQ.ISYMB(MM)) GO TO 1363
1362 CONTINUE
1363 PARAM(3)=MM-1
  READ(5,100,END=60) FRMAX,FRMIN
  GO TO 60
1354 CONTINUE
1357 SENSFL=0
  GO TO 60
3 K=J
  GO TO (67,69,68,68,69),NASA
67 WRITE (6,119)
  WRITE (6,109)
  NASA=4
68 WRITE (6,116)INLST
  GO TO 66
301 GO TO (67,303,305,68,68),NASA
303 WRITE (6,304)

```



```

GOTO 68
305 K=J+1
    K1=80
    K2=6
    DO 400 J=1,23
        ILOOP(J)=0
        MODNM(J)=0
        DO 400 L2=1,10
400 VALMOD(J,L2)=0.
    L1=6
    L2=1
    M9=1
    ISGN=1
    LD=0
    IZ=0
    LI1=0
    IS=1
    WRITE (6,119)
    GOTO 69
4 NASA=2
    K1=30
    K2=0
    WRITE (6,119)
    WRITE (6,116) INLST
    DO 5 I=1,21
        Z(I)=0.
        Z(I+21)=0.
        NODE(I)=0
5 DO 5 J=1,42
    INC(I,J)=0
    MTRAN(1)=0
    MTRAN(2)=0
    MTRAN(3)=0
    MC=0
    N=0
    IPRIN=0
    M=2
    MOVFL=23
6 ISGN=+1
    LD=0
    L1=1
    M=M+1
    IF(M-MOVFL-3) 7,96,96
7 L2=1
    IZ=0
    GO TO 66
69 WRITE (6,116) INLST
    DO 47 J=K,K1
        ILST=ITRAN(INLST(J))
        IF(INSR1(ILST))12,98,10
10 L3=INSR1(ILST)
    ILST=L3
    IF(INSR2(L3,L1))99,98,11
11 L3=INSR2(L3,L1)+1
    GO TO 13
12 L3=1
    ILST=-INSR1(ILST)-1
13 GOTO(14,15,16,17,18,308),L1
14 L3=ITSK1(L3,L2)
    I=1
    GO TO 19
15 L3=ITSK2(L3,L2)
    GO TO 19
16 L3=ITSK3(L3,L2)
    GO TO 19
17 L3=ITSK4(L3,L2)
    I=2
    GO TO 19
18 L3=ITSK5(L3,L2)
    GOTO 19
308 L3=ITASK6(L3,L2)
19 GO TO (20,21,21,24,26,27,28,29,30,31,33,

```



```

+36,37,38,39,40,45,46,47,51,66,97,98,99,41,42,43,44,
+310,311,312,313,314,316,316,319,316,316,316,316,321,
+375),L3
20 MN=10*MN+ILST
21 INC(MN+1,M)=4*(L2-3)**3/((L2-3)**2+1)**2
  IF(MN-N) 23,23,22
22 N=MN
23 NODE(MN)=NODE(MN)+1
  GOTO(46,45,39),L3
24 IF(ILST)99,47,25
25 MN=ILST
  GO TO 46
26 ISGN=-1
  GO TO 46
27 INC(I,M)=100*ISGN*ILST
  GO TO 46
28 INC(I,M)=INC(I,M)+ISGN*MN
  GO TO (39,46),I
29 INC(I,M)=INC(I,M)+ISGN*(10*MN+ILST)
  GO TO 46
30 LD=LD+1
31 IZ=10*IZ+ISGN*ILST
  IF(ILST)99,47,32
32 L2=L2+(120+(L2-K2)*(2-(L2-K2)*6))/36
  GOTO 47
33 IF(INC(1,M)/100-4) 34,35,34
34 LD=LD+15+(ILST-10)*(582-(ILST-10)*(40+(ILST-10)*(22)))
  1/100
35 LD=LD+3-ILST
  IF(L1-6)322,315,99
322 L2=6
  GOTO 47
36 L1=4
  L2=1
  ISGN=(4-ILST)/3
  IF(((INC(1,M)/100)-4)/3)46,98,46
37 ISGN=(4-ILST)/3
  MC=MC+1
  MTRAN(MT)=MD
  MD=MC/2
  ME=MC-2*MD+1
  GOTO 46
38 L1=5
  M2=M-1
  MC=1
  MT=1
  GOTO 7
39 L1=L1+1
  L2=1
  GOTO 47
375 WRITE (6,376)
  NASA=5
40 JC=MC-2*MD+1
  IJTAG(MD,JC)=100*ISGN*ILST
  GOTO 46
41 JC=MC-2*MD+1
  IJTAG(MD,JC)=IJTAG(MD,JC)+ISGN*MN
  GOTO (45,46),ME
42 JC=MC-2*MD+1
  IJTAG(MD,JC)=IJTAG(MD,JC)+ISGN*(10*MN+ILST)
  GOTO 46
43 MT=2
  MTRAN(2)=MTRAN(1)
  GOTO 7
44 MT=3
  MTRAN(3)=MTRAN(1)
  GOTO 7
310 IS=-2
  GOTO(45,46),L2
311 MODNM(M9)=100*ILST
  LI=1
  L2=4

```



```

      GOTO 47
312  MODNM(M9)=MODNM(M9)+10*MN+ILST
      GOTO 46
313  MODNM(M9)=MODNM(M9)+MN
      GOTO 45
319  IS=IS+1
      IF (IS*LI+3) 99,315,324
314  IF (LI-10) 315,315,326
315  V=FLOAT(IZ)
      VALMOD(M9,LI)=V/10.0** (LD)
      L2=12
      IF (IS) 331,99,332
331  IF (LI-3) 47,334,324
334  IF (L3-36) 47,340,47
340  ILOOP(M9)=(VALMOD(M9,1)-VALMOD(M9,2))/VALMOD(M9,3)-1.0
10001
      IF (ILOOP(M9)) 333,328,328
332  ILOOP(M9)=LI
333  IF (LI1-IABS(ILOOP(M9))) 335,47,47
335  LI1=IABS(ILOOP(M9))
      GOTO 47
316  IZ=0
      LD=0
      ISGN=1
      IF (L3-35) 317,317,323
323  LI=LI+1
      L2=7
      L3=L3-36
      GOTO (31,26,45,47),L3
317  IF (IABS(IS)-1) 99,338,324
338  IS=1
      L3=L3-33
      GOTO (320,337),L3
320  IS=-2
      L2=3
337  M9=M9+1
      IF (M9-M) 339,339,326
339  GOTO (47,311),L3
321  IF (MODNM(M9)/100-4) 34,35,34
45  L2=L2+1
46  L2=L2+1
47  CONTINUE
      IF (LI-6) 330,336,99
336  LI=LI1
      IND=4
      IF (IABS(IS)-1) 99,60,324
330  IF ((10*LI+L2)/11-3) 98,48,49
48  INC(2,M)=0
49  I=1
      IF (LI-5) 50,7,99
50  Z(M)=FLOAT(IZ)
      Z(M)=Z(M)/10.0** (LD)
      GOTO 6
51  N1=N+1
      IF (MTRAN(1)) 99,95,202
202  IF (INLST(J+8)-ISYMB(26)) 204,52,204
204  IF (INLST(J+8)-ISYMB(13)) 55,201,55
201  DO 203 J=1,2
      DO 200 I=1,19
      K=IRAY(I,J)
200  IRAY(I,J)=ISYMB(K)
203  WRITE (6,116) (IRAY(I,J),I=1,19)
52  IPRIN=1
55  NA=0
      DO 58 I=1,N,1
      IF (NODE(I)-1) 58,56,57
56  WRITE (6,112) I
57  NA=NA+1
      NODE(NA)=I
58  CONTINUE
      IF (NASA-2) 70,71,70
70  NASA=4

```



```

GO TO 66
71 NASA=3
   IND=2
   IF(N=NA) 99,60,59
59 WRITE (6,111) (NODE(I), I=1,NA,1)
   WRITE (6,110)
   GO TO 60
91 IND=7
60 RETURN
95 WRITE (6,115)
   GO TO 90
96 WRITE (6,114) MOVFL
90 NASA=4
   GOTO 66
97 WRITE (6,106)
   WRITE (6,116) INLST
72 NASA=5
   GO TO 49
98 WRITE (6,108) J
   WRITE (6,116) INLST
   IF(L1-6)72,66,99
99 WRITE (6,107)
   GO TO 90
326 WRITE (6,327)
   GOTO 66
328 WRITE (6,329)
324 WRITE (6,325)
   GOTO 66
100 FORMAT(E11.3,9X,E11.4)
101 FORMAT(80A1)
106 FORMAT(24H FUNCTIONS NOT AVAILABLE)
107 FORMAT(27H MACHINE LANGUAGE BREAKDOWN)
108 FORMAT(32H INPUT CODING ERROR IN COLUMN,13)
109 FORMAT(39H USE HEADING LABELED NASAP PROBLEM)
110 FORMAT(/,1X,39H THE ANALYSIS WILL CONTINUE ON THE ASSUM
1,5HPTION,
2,26H THAT THE LIST IS CORRECT.,/,1X,13H HOWEVER, SKIP
3,9HPING NODE,
4,39H NUMBERS IS NOT A RECOMMENDED PRACTICE.//)
111 FORMAT(1X,41H THE FOLLOWING NODE NUMBERS ARE MENTIONED
1,6H IN THE,
2,12H INPUT LIST ,/,1X,12,19(1H,,12,1X))
112 FORMAT(//,1X,5H NODE ,12,27H HAS ONLY ONE ELEMENT CONNE
111HCTED TO IT.,//)
114 FORMAT(1X,12H NO MORE THAN,13,18H ELEMENTS ALLOWED.)
115 FORMAT(1X,24H THERE IS NO OUTPUT LIST.)
116 FORMAT(1X,80A1)
117 FORMAT(1X,1116)
119 FORMAT(1H1)
304 FORMAT(/,1X,39H MODIFY AND SYMBOLIC SOLUTION MUST FOLLO
1,9HW EXECUTE//)
325 FORMAT(/,1X,39H FOR LOOPS YOU MUST SPECIFY--(START,END,
1,4H INC))
327 FORMAT(/,1X,39H TOO MANY ELEMENTS OR VALUES IN MODIFY S
1,8H TATEMENT)
329 FORMAT(/,1X,39H START GREATER THAN END AND/OR INC TOO L
1,4HARGE)
376 FORMAT (/,1X,27H THE TRANSFER FUNCTION INPUT/
+1X,33H VARIABLE IS A DEPENDENT QUANTITY.)
998 STOP
END

```

```

FUNCTION ITRAN(I)
ITRAN=1+I/16777216
IF(ITRAN)1,2,3
1 ITRAN=ITRAN+63
  RETURN
3 ITRAN=ITRAN-31
  RETURN
2 WRITE(6,4)

```



```

4      RETURN
      FORMAT(//1X,15H MACHINE FAILURE)
      STOP
      END

```

```

      SUBROUTINE NAS4
      INTEGER SENSFL,FLAG,EXPNS,SIGN,WHERE,EXPDS,SIGD,
1PARAM,PARTLN,PARTLD
      DIMENSION JZ(5),AZ(5),LNUM(23,3),LDEN(23,3),ILGTH(23),
1IFMT1(18),IFMT2(12)
      COMMON/SEN/PARAM(3),SENSFL,FRMAX,FRMIN
      COMMON IND,NASA,MTRN1,MTRN2,MTRN3,MD,IJTAG(15,2),I
1PRIN,Z(42),LL,MODNM(23),VALMOD(23,10),ILOOP(23),LI,LIC
2,M9,ZZ(23),NAM(23,3),ITAG(5,8),NMBR(6),
3M,SNUM(30,5),SDEN(30),IOD(30),IO(5),NUM(5)
5,XG(5),IP(30,5),ID,NUMZ,DMY(23)
4,NAME(23,2),MVI,IST(40),LIM(6),ITWO(23),LOOP(10000)
6,WHERE,PARTLN(200,23),EXPNS
7(200),NFN(200),PARTLD(200,23),EXPDS(200),
8SIGD(200),NFD(200),LPN,LPD,SIGN(200)
      DATA ILGTH/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,2H10,
12H11,2H12,2H13,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21
2,2H22,2H23/
      DATA IFMT1/4H(1X,,4H15,I,4H8,3X,4H,A1,,4H1H(,,1H1,
14H(A1,,1H2,4H11),,4H3H/,2H,,1H1,4H(A1,,4H211),
24H,6H),4H S*,4H*,I3,1H)/
      DATA IFMT2/4H(1X,,4H15,I,4H8,3X,4H,A1,,4H1H(,,1H1,
14H(A1,,1H2,4H11),,4H6H),4HS**,4H,I3)/
      DATA IM/'-','/,'IB/' ','/,'AOC/'A1,'/,'ITW/'2 '/'
      MDD=MTRN1+1
      IF(IPRIN-1)1,1,23
1      IERR=0
      IF(LIC)60,2,5
2      DO 3 N1=1,MVI
3      ZZ(N1)=Z(N1)
      IF(ILOOP(1)-1000)5,16,60
16      DO 19 N1=1,MTRN1
      DO 19 L=1,2
19      IJTAG(N1,L)=IJTAG(N1,L)-1000*(IJTAG(N1,L)/1000)
      IF(IPRIN)60,13,23
23      I=0
      LPN=0
      DO 1024 K2=1,MDD
      N7=LIM(K2)
      WHERE=0
      IF(K2-MDD)50,51,60
50      WRITE (6,135) (ITAG(K2,K),K=1,8)
      WRITE (6,160)
      J=-1
      IF(NMBR(K2))601,52,600
600      JS=IM
      GOTO 602
601      JS=IB
602      WRITE (6,162) JS
      GOTO 52
51      WRITE (6,138)
      WRITE (6,160)
      JS=IB
      WRITE (6,162) JS
      SIGD(1)=JS
      NFD(1)=1
      LPD=1
      J=1
52      N8=1
      IF(N7)60,55,56
55      WRITE (6,100)
      WRITE (6,141)
      GO TO 1024
56      DO 24 N9=1,N7

```



```

FLAG=0
KK=0
KKK=0
ISLOP=0
JS=18
I=I+1
LK1=LOOP(I)
LK2=IABS(LK1)
IF(J*LK1)44,60,25
44 JS=IM
25 N1=N8
26 IF(LK2-ITWO(N1))27,28,28
27 IF(N1-N8)60,30,29
30 N8=N8+1
GOTO 25
28 ISLOP=ISLOP+IST(N1)
NM=NAME(N1,1)/100
IF((NM+3)*(NM-4)*(NM-5))32,31,32
31 KKK=KKK+1
DO 230 K=1,3
230 LDEN(KKK,K)=NAM(N1,K)
GO TO 33
32 KK=KK+1
DO 231 K=1,3
231 LNUM(KK,K)=NAM(N1,K)
33 LK2=LK2-ITWO(N1)
29 N1=N1+1
IF(N1-MVI-1)26,499,60
499 IF(SENSFL-1)34,500,60
500 JJ=1
502 IF(K2-MDD)509,799,60
509 LPN=LPN+1
IF(LP.N.GT.200) WRITE(6,164)
IF(KK.EQ.0) GO TO 531
510 JJJ=1
DO 511 K=1,3
IF(K.GT.1) JJJ=4
CALL PACK(PARTLN(LPN,JJ),K,LNUM(JJ,K),JJJ)
511 CONTINUE
IF(LNUM(JJ,1).EQ.PARAM(1).AND.LNUM(JJ,2).EQ.PARAM(2).A
1ND.LNUM(JJ,3).EQ.PARAM(3)) GO TO 519
CALL PACK(PARTLN(LPN,JJ),4,0,4)
GO TO 530
519 FLAG=1
CALL PACK(PARTLN(LPN,JJ),4,3,4)
530 JJ=JJ+1
IF(JJ-KK-1)510,531,60
531 IF(KKK.EQ.0) GO TO 554
LX=0
JJ=JJ-1
534 JJ=JJ+1
LX=LX+1
JJJ=1
DO 532 K=1,3
IF(K.GT.1) JJJ=4
CALL PACK(PARTLN(LPN,JJ),K,LDEN(LX,K),JJJ)
532 CONTINUE
533 IF(LDEN(LX,1).EQ.PARAM(1).AND.LDEN(LX,2).EQ.PARAM(2).A
1ND.LDEN(LX,3).EQ.PARAM(3)) GO TO 540
CALL PACK(PARTLN(LPN,JJ),4,6,4)
GO TO 542
540 FLAG=1
CALL PACK(PARTLN(LPN,JJ),4,7,4)
542 IF(LX-KKK)534,555,60
554 JJ=JJ-1
555 WHERE=1
GO TO 34
799 LPD=LPD+1
IF(LP.D.GT.200) WRITE(6,164)
IF(KK.EQ.0) GO TO 831
800 JJJ=1
DO 812 K=1,3

```



```

      IF(K.GT.1) JJJ=4
      CALL PACK(PARTLD(LPD,JJ),K,LNUM(JJ,K),JJJ)
812  CONTINUE
      IF(LNUM(JJ,1).EQ.PARAM(1).AND.LNUM(JJ,2).EQ.PARAM(2).A
1ND.LNUM(JJ,3).EQ.PARAM(3)) GO TO 819
      CALL PACK(PARTLD(LPD,JJ),4,0,4)
      GO TO 830
819  FLAG=1
      CALL PACK(PARTLD(LPD,JJ),4,3,4)
830  JJ=JJ+1
      IF(JJ-KK-1)800,831,60
831  IF(KKK.EQ.0) GO TO 854
      LX=0
      JJ=JJ-1
834  JJ=JJ+1
      LX=LX+1
      DO 832 K=1,3
      JJJ=1
      IF(K.GT.1) JJJ=4
      CALL PACK(PARTLD(LPD,JJ),K,LDEN(LX,K),JJJ)
832  CONTINUE
833  IF(LDEN(LX,1).EQ.PARAM(1).AND.LDEN(LX,2).EQ.PARAM(2).A
1ND.LDEN(LX,3).EQ.PARAM(3)) GO TO 840
      CALL PACK(PARTLD(LPD,JJ),4,6,4)
      GO TO 842
840  FLAG=1
      CALL PACK(PARTLD(I,JJ),4,7,4)
842  IF(LX-KKK)834,855,60
854  JJ=JJ-1
855  WHERE=2
      34 IF(KK)60,42,43
      42 KK=1
      LNUM(1,1)=IB
      LNUM(1,2)=IB
      LNUM(1,3)=1
      IFMT1(8)=IAOC
      43 IF(KKK)60,40,41
      40 IFMT2(6)=ILGTH(KK)
      WRITE(6,IFMT2)I,LK1,JS,((LNUM(K,K1),K1=1,3),K=1,KK),IS
1LOP
      GO TO 901
      41 IFMT1(6)=ILGTH(KK)
      IFMT1(12)=ILGTH(KKK)
      WRITE(6,IFMT1)I,LK1,JS,((LNUM(K,K1),K1=1,3),K=1,KK),((
1LDEN(K,K1),K1=1,3),K=1,KKK),ISLOP
      IF(IFMT1(8)-ITW)232,901,232
232 IFMT1(8)=ITW
901 IF(WHERE-1)24,902,904
902 EXPNS(LPN)=ISLOP
      SIGN(LPN)=JS
      NFN(LPN)=JJ
      IF(FLAG)60,905,932
904 EXPDS(LPD)=ISLOP
      SIGD(LPD)=JS
      NFD(LPD)=JJ
      IF(FLAG)60,908,932
905 DO 906 K=1,JJ
      ITEST=0
      CALL PACK(ITEST,4,PARTLN(LPN,K),4)
      IF(ITEST-5)960,60,961
960 CALL PACK(PARTLN(LPN,K),4,4,4)
      GO TO 906
961 CALL PACK(PARTLN(LPN,K),4,9,4)
906 CONTINUE
      GO TO 932
908 DO 931 K=1,JJ
      ITEST=0
      CALL PACK(ITEST,4,PARTLD(LPD,K),4)
      IF(ITEST-5)962,60,963
962 CALL PACK(PARTLD(LPD,K),4,4,4)
      GO TO 931
963 CALL PACK(PARTLD(LPD,K),4,9,4)

```



```

931  CONTINUE
932  IF(N9.EQ.N7) GO TO 909
    GO TO 24
909  IF(WHERE-1)24,930,930
930  CALL SENS
24   CONTINUE
1024 CONTINUE
    IPRIN=IPRIN-1
    IF(IPRIN)13,13,89
5    LIC=LIC+1
    WRITE (6,163)
    DO 12 N1=1,M9
    IF(LIC-IABS(ILOOP(N1)))4,4,12
4    DO 11 N7=1,MVI
    IF(MODNM(N1)-IABS(NAME(N7,1)))11,6,11
6    IF(ILOOP(N1))8,60,7
7    ZZ(N7)=VALMOD(N1,LIC)
    GOTO 9
8    ZZ(N7)=VALMOD(N1,1)+FLOAT(LIC-1)*VALMOD(N1,3)
9    NM=NAME(N7,1)/100
    WRITE (6,129) (NAM(N7,K),K=1,3),ZZ(N7)
    IF((NM+3)*(NM-4)*(NM-5))12,10,12
10   ZZ(N7)=1./ZZ(N7)
    GOTO 12
11   CONTINUE
    WRITE (6,130)
    IERR=1
    MVI1=MVI+1
    DO 15 N7=MVI1,M
    IF(MODNM(N1)-IABS(NAME(N7,1)))15,14,15
14   WRITE (6,133) MODNM(N1)
    GOTO 12
15   CONTINUE
    WRITE (6,132) MODNM(N1)
12   CONTINUE
    IF(IERR)60,13,66
13   IF(ILOOP(1)-1000)17,18,60
18   WRITE (6,101)
17   DO 400 N1=1,30
    SDEN(N1)=0.
    DO 400 K2=1,MTRN1
400   SNUM(N1,K2)=0.
    DO 603 K2=1,MTRN1
603   SNUM(16,K2)=NMBR(K2)
    SDEN(16)=1.
    I=0
    DO 1412 K2=1,MDD
    N8=1
    N7=LIM(K2)
    IF(N7)60,1412,397
397  DO 412 N9=1,N7
    I=I+1
    ISLOP=0
    VLOOP=1.
    LK1=LOOP(I)
    LK2=IABS(LK1)
    SGN=LK1/LK2
401  N1=N8
402  IF(LK2-ITWO(N1)) 403,405,405
403  IF(N1-N8)60,404,406
404  N8=N8+1
    GOTO 401
405  ISLOP=ISLOP+IST(N1)
    VLOOP=VLOOP*ZZ(N1)
    LK2=LK2-ITWO(N1)
406  N1=N1+1
    IF(N1-MVI-1)402,407,60
407  JS=16-ISLOP
    IF(MDD-K2) 60,410,408
408  SNUM(JS,K2)=SNUM(JS,K2)+VLOOP*SGN
    GOTO 412
410  SDEN(JS)=SDEN(JS)+VLOOP*SGN

```



```

412 CONTINUE
1412 CONTINUE
DO 69 K=1,MTRN1
  JZ(K)=1
  AZ(K)=0.
  DO 68 J=1,30
    IF(SNUM(J,K))67,68,67
67 IF(AZ(K))201,202,201
202 AZ(K)=-SNUM(J,K)
201 JZ(K)=J
68 CONTINUE
69 CONTINUE
  SIGNC=0.
  JI=0
  DO 71 J=1,30
    IF(SDEN(J))97,71,97
97 IF(SIGNC)70,98,70
98 SIGNC=SDEN(J)
70 JI=J
71 CONTINUE
  J=0
  DO 73 K=1,MTRN1
    IF(JZ(K)-J)73,73,72
72 J=JZ(K)
73 CONTINUE
    IF(JI-J)75,75,74
74 J=JI
75 DO 83 K=1,MTRN1
    XG(K)=0.
    IF(SIGNC)90,91,90
90 XG(K)=AZ(K)/SIGNC
91 WRITE (6,136) XG(K)
    WRITE (6,135) (ITAG(K,I),I=1,8)
    NU=0
    IO(K)=0
    DO 82 L=1,J
      ANUM=NU
      IF(ABS(SNUM(L,K))*(ANUM-.5))80,82,81
80 IO(K)=J-L
81 NU=NU+1
      IP(NU,K)=J-L
      SNUM(NU,K)=-SNUM(L,K)/AZ(K)
      WRITE (6,137) SNUM(NU,K),IP(NU,K)
82 CONTINUE
    NUM(K)=NU
    IF(NUM(K))60,92,83
92 NUM(K)=1
    WRITE (6,137) SNUM(1,K),IP(1,K)
    WRITE (6,141)
83 CONTINUE
    WRITE (6,138)
    NUMZ=0
    ID=0
    DO 86 K=1,J
      ANUM=NUMZ
      IF(ABS(SDEN(K))*(ANUM-.5))84,86,85
84 ID=J-K
85 NUMZ=NUMZ+1
      IOD(NUMZ)=J-K
      SDEN(NUMZ)=SDEN(K)/SIGNC
      WRITE (6,137) SDEN(NUMZ),IOD(NUMZ)
86 CONTINUE
      IF(NUMZ)60,94,87
94 NUMZ=1
      WRITE (6,137) SDEN(1),IOD(1)
      WRITE (6,141)
87 IF(LI-LIC)150,150,152
150 LIC=0
152 IF(MTRN1-MTRN2)96,96,95
95 IF(MTRN1-MTRN3)88,88,153
153 IF(LIC)60,89,5
96 IND=5

```



```

      GO TO 900
88    IND=6
      GO TO 900
60    WRITE (6,134)
66    LIC=0
89    IND=1
900   RETURN
100   FORMAT(//1X,16HIDENTICALLY ZERO)
101   FORMAT(//1X,16HNOMINAL SOLUTION)
129   FORMAT(/1X,10HMODIFYING ,A1,2I1,4H TO ,E11.4)
130   FORMAT(//1X,26HERROR IN ATTEMPT TO MODIFY)
132   FORMAT(1X,12HBRANCH CODED,I5,22H--NOT IN ORIGINAL NETW
1,3HORK)
133   FORMAT(1X,12HBRANCH CODED,I5,22H--CANNOT MODIFY INDEPE
1,13HNDET SOURCES)
134   FORMAT(//1X,24HMACHINE LANGUAGE FAILURE)
135   FORMAT(/1X,24HNUMERATOR POLYNOMIAL OF ,2A1,2I1,1H/,2A1
1,2I1)
136   FORMAT(//1X,4HK = ,E11.4)
137   FORMAT(1X,E11.4,10H TIMES S**,I3)
138   FORMAT(//1X,22HDENOMINATOR POLYNOMIAL)
141   FORMAT(/,1X,31HCHECK YOUR CIRCUIT CODING LIST.,
+/,1X,26HTHIS ANSWER IS DEGENERATE.)
160   FORMAT(/3X,28HLOC CODE LOOP PRODUCT)
162   FORMAT(17X,A1,13H( 1 ) S ** 0)
163   FORMAT(/)
164   FORMAT(' ', 'WARNING: NUMBER OF UNCANCELLED LOOP PRODU'
1, 'CTS EXCEEDS STORAGE AVAILABLE (200). ', /' ', 'YOU MU'
2, 'ST CHANGE DIMENSION STATEMENTS IN NAS4 AND SENS SUB'
3, 'ROUTINES FOR VARIABLES - ', /' ', 'PARTLN, EXPNS, NFN'
4, ' ', PARTLD, EXPDS, SIGD, NFD, SIGN, RNUMP, RNUMS, RDENS, RDENP'
5)
      STOP
      END

```

C PACKING SUBROUTINE

```

SUBROUTINE PACK(A,I,B,J)
LOGICAL*1 A(1),B(1)
A(I)=B(J)
RETURN
END

```

```

SUBROUTINE SENS
COMPLEX*8 CFREQ,CTEMP,CTEMP1,CTEMPB
INTEGER EXPNS,SIGN,EXPDS,SIGD,WHERE,PARAM,PARTLD,
1PARTLN
DIMENSION RNUMP(200),RNUMS(200),RDENS(200),RDENP(200),
1ITEMP(23),IEXPN(50),IEXPD(50),PART1(50),PART2(50)
COMMON/SEN/PARAM(3),SENSFL,FRMAX,FRMIN
COMMON IND,NASA,MTRN1,MTRN2,MTRN3,MD,IJTAG(15,2),I
1PRIN,Z(42),LL,MODNM(23),VALMOD(23,10),ILOOP(23),LI,LIC
2,M9,ZZ(23),NAM(23,3),ITAG(5,8),NMBR(6),
3M,SNUM(30,5),SDEN(30),IOD(30),IO(5),NUM(5)
5,XG(5),IP(30,5),ID,NUMZ,DMY(23)
4,NAME(23,2),MVI,IST(40),LIM(6),ITWO(23),LOOP(10000)
6,WHERE,PARTLN(200,23),EXPNS
7(200),NFN(200),PARTLD(200,23),EXPDS(200),
8SIGD(200),NFD(200),LPN,LPD,SIGN(200)
DATA INEG/'- ' /,IBLNK/' ' /,PART1/50*0.0/,PART2/50
1*0.0/
DO 2 K=1,MVI
ITEMP(K)=NAM(K,1)
DO 2 L=2,4
CALL PACK(ITEMP(K),L,0,4)
2 CONTINUE
IF(WHERE-1)60,200,19

```



```

19  CALL PACK(PARTLD(1,1),3,1,4)
    CALL PACK(PARTLD(1,1),4,4,4)
    NFD(1)=1
    GO TO 400
200  DO 320 I=1,LPN
    RNUMP(I)=1
    RNUMS(I)=1
    KP=NFN(I)
    DO 320 J=1,KP
    ICOMP1=0
    ICOMP2=0
    ICOMP3=0
    ICOMP4=0
    CALL PACK(ICOMP1,1,PARTLN(I,J),1)
    CALL PACK(ICOMP2,4,PARTLN(I,J),2)
    CALL PACK(ICOMP3,4,PARTLN(I,J),3)
    KL=0
205  KL=KL+1
    IF(ICOMP1.EQ.ITEMP(KL)) GO TO 210
    GO TO 225
210  IF(ICOMP2.EQ.NAM(KL,2)) GO TO 220
    GO TO 225
220  IF(ICOMP3.EQ.NAM(KL,3)) GO TO 240
225  IF(KL.LT.MVI) GO TO 205
240  CALL PACK(ICOMP4,4,PARTLN(I,J),4)
    ICOMP4=ICOMP4+1
    GO TO (250,60,60,285,280,60,290,300,60,310),ICOMP4
250  RNUMP(I)=RNUMP(I)/Z(KL)
    GO TO 285
280  RNUMP(I)=0
285  RNUMS(I)=RNUMS(I)/Z(KL)
    GO TO 316
290  RNUMP(I)=RNUMP(I)*Z(KL)
    GO TO 315
300  RNUMP(I)=RNUMP(I)*Z(KL)**2
    RNUMP(I)=-RNUMP(I)
    GO TO 315
310  RNUMP(I)=0
315  RNUMS(I)=RNUMS(I)*Z(KL)
316  IF(SIGN(I).EQ.INEG) GO TO 317
    GO TO 320
317  RNUMP(I)=-RNUMP(I)
    RNUMS(I)=-RNUMS(I)
320  CONTINUE
    GO TO 30
400  DO 2520 I=1,LPD
    KP=NFD(I)
    RDENP(I)=1
    RDENS(I)=1
    IF(I.NE.1) GO TO 402
    RDENP(I)=0
    GO TO 2520
402  DO 520 J=1,KP
    ICOMP1=0
    ICOMP2=0
    ICOMP3=0
    ICOMP4=0
    CALL PACK(ICOMP1,1,PARTLD(I,J),1)
    CALL PACK(ICOMP2,4,PARTLD(I,J),2)
    CALL PACK(ICOMP3,4,PARTLD(I,J),3)
    KL=0
405  KL=KL+1
    IF(ICOMP1.EQ.ITEMP(KL)) GO TO 410
    GO TO 425
410  IF(ICOMP2.EQ.NAM(KL,2)) GO TO 420
    GO TO 425
420  IF(ICOMP3.EQ.NAM(KL,3)) GO TO 440
425  IF(KL.LT.MVI) GO TO 405
440  CALL PACK(ICOMP4,4,PARTLD(I,J),4)
    ICOMP4=ICOMP4+1
    GO TO (450,60,60,485,480,60,490,500,60,510),ICOMP4
450  RDENP(I)=RDENP(I)/Z(KL)

```



```

GO TO 485
480 RDENP(I)=0.0
485 RDENS(I)=RDENS(I)/Z(KL)
GO TO 516
490 RDENP(I)=RDENP(I)*Z(KL)
GO TO 515
500 RDENP(I)=RDENP(I)*Z(KL)**2
RDENP(I)=-RDENP(I)
GO TO 515
510 RDENP(I)=0.0
515 RDENS(I)=RDENS(I)*Z(KL)
516 IF(SIGD(I).EQ.INEG) GO TO 517
GO TO 520
517 RDENP(I)=-RDENP(I)
RDENS(I)=-RDENS(I)
520 CONTINUE
2520 CONTINUE
J=0
DO 610 K=1,LPD
DO 610 I=1,LPN
J=J+1
IEXPN(J)=EXPNS(I)+EXPDS(K)
IF(ABS(RDENS(K)*RNUMP(I)-RNUMS(I)*RDENP(K)).LE.1.0E-75
1) GO TO 1520
GO TO 1521
1520 J=J-1
GO TO 610
1521 IF(J.EQ.1) GO TO 1620
JMIN1=J-1
DO 1610 KL=1,JMIN1
IF(IEXPN(J).EQ.IEXPN(KL)) GO TO 608
GO TO 1610
608 PART1(KL)=PART1(KL)+RDENS(K)*RNUMP(I)-RNUMS(I)*RDENP(K)
1)
J=J-1
GO TO 610
1610 CONTINUE
1620 PART1(J)=RDENS(K)*RNUMP(I)-RNUMS(I)*RDENP(K)
610 CONTINUE
IF(J.LE.1) GO TO 640
JJ=1
631 K=JJ+1
IF(ABS(PART1(JJ)).LE.1.0E-75) GO TO 636
632 IF(IEXPN(JJ).GT.IEXPN(K)) GO TO 634
TEMP1=PART1(JJ)
TEMP2=IEXPN(JJ)
PART1(JJ)=PART1(K)
IEXPN(JJ)=IEXPN(K)
PART1(K)=TEMP1
IEXPN(K)=TEMP2
634 K=K+1
IF(K.GT.J) GO TO 636
GO TO 632
636 IF(JJ.EQ.(J-1)) GO TO 640
JJ=JJ+1
GO TO 631
638 WRITE(6,900) (ITAG(1,K),K=1,8),(PARAM(K),K=1,3)
IF(J)60,641,642
641 WRITE(6,906)
GO TO 643
642 SET=0
DO 710 K=1,J
IF(ABS(PART1(K)).LT.1.0E-75) GO TO 710
SET=1
700 WRITE(6,901) PART1(K),IEXPN(K)
710 CONTINUE
IF(SET.EQ.0) WRITE(6,906)
643 J1=J
J=0
DO 611 K=1,LPD
DO 611 I=1,LPD
J=J+1

```



```

      IEXPDP(J)=EXPDS(I)+EXPDS(K)
      IF(ABS(RDENS(K)*RDENS(I)).LE.1.0E-75) GO TO 1711
      GO TO 1712
1711  J=J-1
      GO TO 611
1712  IF(J.EQ.1) GO TO 1612
      JMIN1=J-1
      DO 1611 KL=1,JMIN1
      IF(IEXPDP(J).EQ.IEXPDP(KL)) GO TO 609
      GO TO 1611
609   PART2(KL)=PART2(KL)+RDENS(K)*RDENS(I)
      J=J-1
      GO TO 611
1611  CONTINUE
1612  PART2(J)=RDENS(K)*RDENS(I)
611   CONTINUE
      IF(J-1)60,749,740
740   JJ=1
741   K=JJ+1
      IF(ABS(PART2(JJ)).LE.1.0E-75) GO TO 746
742   IF(IEXPDP(JJ).GT.IEXPDP(K)) GO TO 744
      TEMP1=PART2(JJ)
      TEMP2=IEXPDP(JJ)
      PART2(JJ)=PART2(K)
      IEXPDP(JJ)=IEXPDP(K)
      PART2(K)=TEMP1
      IEXPDP(K)=TEMP2
744   K=K+1
      IF(K.GT.J) GO TO 746
      GO TO 742
746   IF(JJ.EQ.(J-1)) GO TO 749
      JJ=JJ+1
748   GO TO 741
749   WRITE(6,902)
      DO 750 K=1,J
      IF(ABS(PART2(K)).LT.1.0E-75) GO TO 750
      WRITE(6,901) PART2(K),IEXPDP(K)
750   CONTINUE
      IF(FRMAX.EQ.0.0.AND.FRMIN.EQ.0.0) GO TO 30
      FRINC=(FRMAX-FRMIN)/199.0
      FREQ=FRMIN
      WRITE(6,904)
      DO 763 KK=1,200
      FREQ1=2.0*3.1415926535*FREQ
      CFREQ=CMPLX(0.0,FREQ1)
      CTEMP=(0.0,0.0)
      DO 755 K=1,J1
      CTEMP1=CMPLX(PART1(K),0.0)
      IF(IEXPDP(K).LT.0) GO TO 754
      CTEMP=CTEMP+CTEMP1*CFREQ**IEXPDP(K)
      GO TO 755
754   CTEMP=CTEMP+CTEMP1/CFREQ**(IABS(IEXPDP(K)))
755   CONTINUE
      CTEMPB=(0.0,0.0)
      DO 760 K=1,J
      CTEMP1=CMPLX(PART2(K),0.0)
      IF(IEXPDP(K).LT.0) GO TO 759
      CTEMPB=CTEMPB+CTEMP1*CFREQ**IEXPDP(K)
      GO TO 760
759   CTEMPB=CTEMPB+CTEMP1/CFREQ**(IABS(IEXPDP(K)))
760   CONTINUE
      IF(CABS(CTEMPB).LT.1.0E-74) GO TO 1762
      CTEMP=CTEMP/CTEMPB
      RMAGN=CABS(CTEMP)
      ANGLE=ATAN2(AIMAG(CTEMP),REAL(CTEMP))
762   WRITE(6,905) FREQ,RMAGN,ANGLE
1762  FREQ=FREQ+FRINC
763   CONTINUE
      GO TO 30
900   FORMAT(' ',/1X,31HNUMERATOR POLYNOMIAL : SENSITIV
1,'ITY OF ',2A1,2I1,
21H/,2A1,2I1,2X,4HTO ',A1,2I1)

```



```

901  FORMAT(' ',/,1X,E11.4,10H TIMES S**,I3)
902  FORMAT(' ',////,1X,27HDENOMINATOR POLYNOMIAL :  S
1,'ENSITIVITY')
903  FORMAT(' ',//,' ', 'MACHINE FAILURE')
904  FORMAT(' ',////,' ', 'EVALUATION OF SENSITIVITY',
1//,' ', 'FREQUENCY'
2,10X,'MAGNITUDE',10X,'PHASE(IN RADIAN)',/, ' ')
905  FORMAT(' ',E11.4,8X,E11.4,8X,E11.4)
906  FORMAT(' ',//,' ', 'IDENTICALLY ZERO',//,' ')
60  WRITE(6,903)
30  RETURN
END

```


BIBLIOGRAPHY

1. Bach, R. E. Jr., Schwartz, R. S. and Daniel, J. H., NASAP User's Guide, local publication of the Electrical Engineering Department, U. S. Naval Postgraduate School, Monterey, California, July 1972.
2. Bach, R. E. Jr., Schwartz, R. S. and Daniel, J. H., NASAP Programmer's Guide, local publication of the Electrical Engineering Department, U. S. Naval Postgraduate School, Monterey, California, July 1972.
3. Bach, R. E. Jr. and Schwartz, R. S., Symbolic Representation of Network Solutions, paper presented at Purdue University Symposium, 1971.
4. Butler, E. M., "Realistic Design Using Large-Change Sensitivities and Performance Contours," IEEE Transactions on Circuit Theory, v. CT-18, p. 58-65, January 1971.
5. Calahan, D. A., "Computer Design of Linear Frequency Selective Networks," Proceedings of the IEEE, v. 53, No. 11, p. 1701-1706, November 1965.
6. Chan, Shu-Park, The Role of Topological Analysis in Computer-Aided Network Design, paper presented at the Mexico International IEEE Conference on Systems, Networks and Computers, Oaxtepec, Mor., Mexico, 19-21 January 1971.
7. Chan, Shu-Park, Topological Analysis: The Flowgraph Approach Versus the K-Tree Approach, paper presented at the IEEE Circuit Theory Group CAD Seminar, Palo Alto, California, 14 November 1970.
8. Chan, Shu-Park, An Introduction to Topological Methods in the Analysis of Passive Networks, local publication of the University of Santa Clara, California, 1970.
9. General Dynamics Corporation, NASA Contractor Report NASA CR-824, Analysis and Design of Space Vehicle Flight Control Systems, Volume V - Sensitivity Theory, by A. L. Greensite, July 1967.
10. Director, S. W., Automated Network Design: Principles and Techniques, Ph.D. Thesis, University of California, Berkeley, 1969.
11. Director, S. W., "Automated Network Design - The Frequency Domain Case," IEEE Transactions on Circuit Theory, v. CT-16, No. 3, p. 330-337, August 1969.
12. Director, S. W., "Survey of Circuit-Oriented Optimization Techniques," IEEE Transactions on Circuit Theory, v. CT-18 No. 1, p. 3-10, January 1971.

13. Fletcher, R. and Powell, M.J.D., "A Rapidly Convergent Descent Method for Minimization," The Computer Journal, v. 6, No. 2, p. 163-168, July 1963.
14. Fletcher, R. and Reeves, C. M., "Function Minimization by Conjugate Gradients," The Computer Journal, v. 7, No. 2, p. 149-153, July 1964.
15. McCalla, W.J. and Pederson, D.O., "Elements of Computer-Aided Circuit Analysis," IEEE Transactions on Circuit Theory, v. CT-18, No. 1, p. 14-25, January 1971.
16. University of California, Los Angeles Report UCLA-ENG-7044, NASAP-70 User's and Programmer's Manual, by H. Okrent and L.P. McNamee, October 1970.
17. Parker, S.R., "Sensitivity: Old Questions, Some New Answers," IEEE Transactions on Circuit Theory, v. CT-18, No. 1, p. 27-35, January 1971.
18. Parker, S.R., A Comparative Study of Some Sensitivity Analysis Techniques, paper presented at the Fourth Asilomar Conference on Circuits and Systems, November 1970.
19. Rohrer, R.A., "Fully Automated Network Design by Digital Computer: Preliminary Considerations", Proceedings of the IEEE, v. 55, No. 11, p. 1929-1939, November 1967.
20. Schwartz, R.S., A Symbolic Solution Algorithm for Design of Lumped, Linear Electrical Networks, Ph. D. Thesis, Northeastern University, Boston, Massachusetts, May 1972.
21. Temes, G.C. and Calahan, D.A., "Computer-Aided Network Optimization, The State of the Art," Proceedings of the IEEE, v. 55, No. 11, p. 1832-1862, November 1967.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 021Z Naval Postgraduate School Monterey, California 93940	2
3. Professor Shugar Chan, Code 52CD (Thesis Advisor) Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
4. Professor Ralph E. Bach, Jr. (Thesis Advisor) 307 Dana Research Center Northeastern University 300 Huntington Ave. Boston, Massachusetts 02115	1
5. Lieutenant John Hale Daniel, USN P. O. Box 426 Ozona, Florida 33560	1

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY (Corporate author)

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

REPORT TITLE

A Symbolic Sensitivity Computation Algorithm

DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Master's; thesis

AUTHOR(S) (First name, middle initial, last name)

John Hale Daniel

REPORT DATE

December 1972

7a. TOTAL NO. OF PAGES

57

7b. NO. OF REFS

21

8. CONTRACT OR GRANT NO.

9a. ORIGINATOR'S REPORT NUMBER(S)

b. PROJECT NO.

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned
this report)

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Naval Postgraduate School
Monterey, California 93940

13. ABSTRACT

A study and pursuant development is described of a digital computer program for the computation of transfer function sensitivities in a symbolic form suitable for storage and subsequent repetitive numerical evaluation over a range of frequencies or parameter values. The algorithm is implemented in conjunction with the Network Analysis for Systems Applications Program (NASAP) developed by R. S. Schwartz at Northeastern University. Several example problems, as well as suggestions for possible improvement of the algorithm and its extrapolation into the areas of optimization and automatic design, are included.

14.

KEY WORDS

LINK A

LINK B

LINK C

ROLE

WT

ROLE

WT

ROLE

WT

Sensitivity

Sensitivity Algorithm

Sensitivity Computation Program

Circuit Analysis-Sensitivity

28 MAR 78

24654

Thesis
D14775
c.1

Daniel

A symbolic sensitivity
computation algorithm.

141452

28 MAR 79

24654

141452

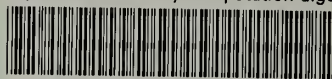
Thesis
D14775
c.1

Daniel

A symbolic sensitivity
computation algorithm.

thesD14775

A symbolic sensitivity computation algor



3 2768 001 02297 3

DUDLEY KNOX LIBRARY